Solving the Battleship puzzle as an integer programming problem

Meuffels, W.J.M.; den Hertog, D.

Published in:
INFORMS Transactions on Education

Publication date:
2010

Vol. 10, No. 3, May 2010, pp. 156–162
ISSN 1532-0545 | 10 | 1003 | 0156
INFORMS Transactions on Education
DOI 10.1287/ited.1100.0047

Puzzle

Solving the *Battleship* Puzzle as an Integer Programming Problem

W. J. M. Meuffels
Tilburg University, 5000 LE Tilburg, The Netherlands, and ORTEC, 2800 AL Gouda, The Netherlands, ineke.meuffels@ortec.com
D. den Hertog
Tilburg University, 5000 LE Tilburg, The Netherlands, d.denHertog@uvt.nl

One's aim in solving logical puzzles is to find the solution by making use of several clues and restrictions. In this paper, we solve a logical puzzle, the *Battleship* puzzle, by integer programming. Moreover, two integer programming models (i.e., a cell-based model and a ship-based model) for the *Battleship* puzzle are compared based on their complexity and solution times. The ship-based model requires more preprocessing work before running the integer program than the cell-based model, but strongly outperforms the latter one. Finally, the models are used to check if a puzzle contains redundant information and to create a puzzle with a unique solution.

1. Introduction

One's aim in solving logical puzzles is to find the solution by making use of several clues and restrictions. Several techniques have been used to solve riddles and puzzles. For example, iterative algorithms are used to solve the *Hanoi* or *Reves* puzzles (Gedeon 1996, Majumdar 1994, Sapir 2004, Sniedovich 2002). Optimal algorithms for *Mastermind* are considered in Chen and Lin (2004). Integer linear programming is used for compiling crossword puzzles (Wilson 1989), *Su Doku* and the *Log Pile* (Chlond 2005), *Rummikub* (Den Hertog and Hulshof 2006), the *n-Queens* problem (Letavec and Ruggiero 2002), and several others. In this paper, integer programming is used to solve the *Battleship* puzzle.

The *Battleship* puzzle, sometimes called *Solitaire Battleships* or *Battleship Solitaire*, is a logical puzzle based on the *Battleship* guessing game. Such puzzles often appear in puzzle magazines. According to Wikipedia (*Battleship* puzzle 2009), the game was invented in Argentina by Jaime Poniachik. The first *Battleship* puzzle was published in 1982 in the Spanish magazine *Humor and Juegos*. After its international debut at the first World Puzzle Championship in New York City (Van de Liefvoort 1992), the game gained more attention and appears regularly in puzzle magazines.

The goal of the *Battleship* puzzle is to determine where the ships are located in a grid. In most puzzle magazines, the sizes of these grids differ, where the smallest puzzle is of size 5×5 and the largest puzzle is of size 15×10. To solve a puzzle, various clues and restrictions are given. There are five sort of ships that can be placed: a submarine of size one, a destroyer of size two, a cruiser of size three, a battleship of size four, and a carrier of size five (Table 1). A first clue that is given, is the number of ships of each sort that has to be placed. The ships are restricted to be placed horizontally or vertically. Further restrictions are that the ships cannot overlap, i.e., at most one ship can occupy any given square in the grid, and that the ships might not touch each other, i.e., each ship must be surrounded by water. A second clue yields information about the number of cells that hide a piece of a ship in a certain column or row. Furthermore, some cells of the initial grid may contain a piece of a ship or water.

Example 1. First note that the information about the number of cells that hide a piece of a ship in a row is given on the right of the grid; the information about the number of cells that hide a piece of a ship in a column is given on the bottom of the grid. Table 1 indicates the number of ships of each sort that has to be placed in the grid of Figure 1.

Table 1 Specification of the Number of Ships of Each Sort That Has to Be Placed in the Grid of Example 1

Type of ship	Symbol	Number to be placed
Submarine	●	5
Destroyer	◀ ▶	4
Cruiser	◀ ■ ▶	3
Battleship	◀ ■ ■ ▶	2
Carrier	◀ ■ ■ ■ ▶	1

By using the various clues and restrictions, Example 1 can be solved. The solution can be found in Figure 2.

In §2, a cell-based model is derived to solve the *Battleship* puzzle. Afterwards, a ship-based model is derived in §3. Section 4 compares the cell-based model and the ship-based model. Finally, a puzzle with a unique solution is created in §5.

2. Cell-Based Model

First the problem is modelled by looking at each specific cell. Each cell may obtain exactly one out of seven symbols; namely, ≈, ◀, ▶, ▲, ▼, ■, or ●. Which symbol corresponds to a cell can be obtained by using the clues and restrictions.

In this section, a definition of sets, parameters, and variables is given. Afterwards, these are used to formulate the cell-based model.

Sets

- *Rows* consists of all rows, where each row is numbered and a particular row is denoted by i, $i \in \{1, \ldots, I\}$; I denotes the last row.

Figure 1 Initial Grid of Example 1

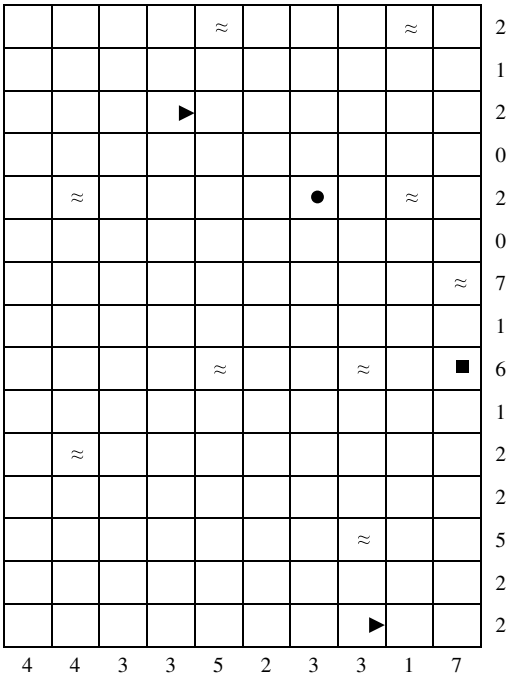
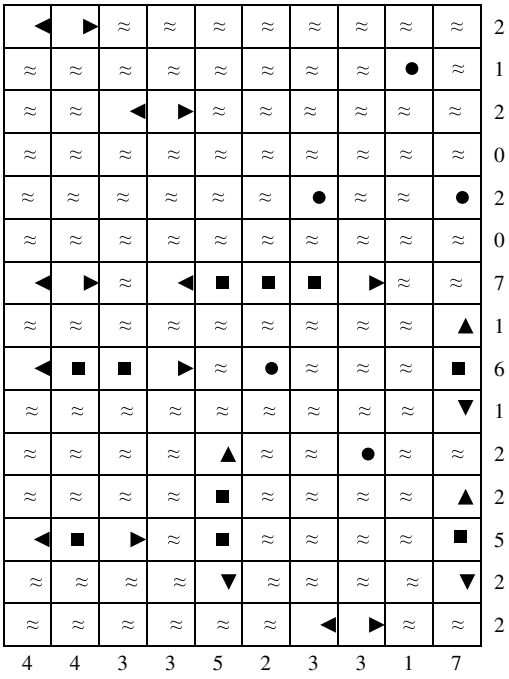


Figure 2 Solution to Example 1



- *Columns* consists of all columns, where each column is numbered and a particular column is denoted by j, $j \in \{1, \ldots, J\}$; J denotes the last column.
- *Celltypes* consists of all seven symbols, where each symbol is numbered and a particular symbol is denoted by k, $k \in \{0, \ldots, 6\}$ and

$$\begin{aligned} k = 0 &\quad \approx \\ 1 &\quad \blacktriangleleft \\ 2 &\quad \blacktriangleright \\ 3 &\quad \blacktriangle \\ 4 &\quad \blacktriangledown \\ 5 &\quad \blacksquare \\ 6 &\quad \bullet. \end{aligned}$$

- *Ships* consists of all ships, where each ship is numbered and each number corresponds to the length of the ship. A particular ship of length l is denoted by l, $l \in \{1, \ldots, 5\}$.

Parameters

- R_i denotes the number of cells that must contain a piece of a ship in row i, $i \in \{1, \ldots, I\}$.
- C_j denotes the number of cells that must contain a piece of a ship in column j, $j \in \{1, \ldots, J\}$.
- $T_{i,j,k} = 1$; if cell (i, j) contains symbol k in the initial grid;
  $= 100$, otherwise.
- S_l denotes the number of ships that has to be placed in the grid for each sort $l \in \{1, \ldots, 5\}$.

Variables

- The decision variable

$$x_{i,j,k} = 1 \quad \text{if cell } (i,j) \text{ contains symbol } k,$$
$$= 0 \quad \text{otherwise.}$$

- The decision variable

$$s_{i,j,l} = 1 \quad \text{if a ship of length } l \text{ starts (i.e., has symbol } \blacktriangleleft, \blacktriangle, \text{ or } \bullet) \text{ in cell } (i,j),$$
$$= 0 \quad \text{otherwise.}$$

Now a cell-based model can be formulated. Notice that the *Battleship* puzzle formulated as an integer programming problem is a so-called feasibility problem. That is, the aim of the model is to find a solution in which all constraints are satisfied.

Constraints

$$\sum_k x_{i,j,k} = 1 \quad \forall i,j, \tag{1}$$

$$x_{i,j,k} = T_{i,j,k} \quad \forall i,j,k \mid T_{i,j,k} = 1, \tag{2}$$

$$\sum_j \sum_{k \mid k \neq 0} x_{i,j,k} = R_i \quad \forall i, \tag{3}$$

$$\sum_j \sum_{k \mid k \neq 0} x_{i,j,k} = C_i \quad \forall j, \tag{4}$$

$$s_{i,j,1} = x_{i,j,6} \quad \forall i,j, \tag{5}$$

$$ls_{i,j,l} \leq x_{i,j,1} + x_{i,j+l-1,2} + \sum_{c=j+1}^{j+l-2} x_{i,c,5} + x_{i,j,3} + x_{i+l-1,j,4} + \sum_{r=i+1}^{i+l-2} x_{r,j,5} \quad \forall i,j,l \mid l \neq 1, \tag{6}$$

$$\sum_i \sum_j s_{i,j,l} = S_l \quad \forall l, \tag{7}$$

$$\sum_{k \mid k \neq 0} x_{i,j,k} + \sum_{k \mid k \neq 0} x_{i+1,j+1,k} \leq 1 \quad \forall i,j, \tag{8}$$

$$\sum_{k \mid k \neq 0} x_{i,j,k} + \sum_{k \mid k \neq 0} x_{i-1,j+1,k} \leq 1 \quad \forall i,j, \tag{9}$$

$$x_{i,j,1} \leq x_{i,j+1,2} + x_{i,j+1,5} \quad \forall i,j, \tag{10}$$

$$x_{i,j,1} \leq x_{i,j-1,0} \quad \forall i,j \mid j \neq 1, \tag{11}$$

$$x_{i,j,2} \leq x_{i,j+1,0} \quad \forall i,j \mid j \neq J, \tag{12}$$

$$x_{i,j,2} \leq x_{i,j-1,1} + x_{i,j-1,5} \quad \forall i,j, \tag{13}$$

$$x_{i,j,3} \leq x_{i+1,j,4} + x_{i+1,j,5} \quad \forall i,j, \tag{14}$$

$$x_{i,j,3} \leq x_{i-1,j,0} \quad \forall i,j \mid i \neq 1, \tag{15}$$

$$x_{i,j,4} \leq x_{i+1,j,0} \quad \forall i,j \mid i \neq I, \tag{16}$$

$$x_{i,j,4} \leq x_{i-1,j,3} + x_{i-1,j,5} \quad \forall i,j, \tag{17}$$

$$x_{i,j,5} \leq x_{i-1,j,3} + x_{i-1,j,5} + x_{i,j+1,2} + x_{i,j+1,5} \quad \forall i,j, \tag{18}$$

$$x_{i,j,5} \leq x_{i-1,j,3} + x_{i-1,j,5} + x_{i,j-1,1} + x_{i,j-1,5} \quad \forall i,j, \tag{19}$$

$$x_{i,j,5} \leq x_{i+1,j,4} + x_{i+1,j,5} + x_{i,j-1,1} + x_{i,j-1,5} \quad \forall i,j, \tag{20}$$

$$x_{i,j,5} \leq x_{i+1,j,4} + x_{i+1,j,5} + x_{i,j+1,2} + x_{i,j+1,5} \quad \forall i,j, \tag{21}$$

$$x_{i,j,6} \leq 1 - x_{i,j-1,6} \quad \forall i,j \mid j \neq 1, \tag{22}$$

$$x_{i,j,6} \leq 1 - x_{i-1,j,6} \quad \forall i,j \mid i \neq 1, \tag{23}$$

$$x_{i,j,k} \in \{0,1\} \quad \forall i,j,k, \tag{24}$$

$$s_{i,j,l} \in \{0,1\} \quad \forall i,j,l. \tag{25}$$

Remark 1. Constraints (6), (8)–(10), (13)–(14), and (17)–(21) may refer to cells that are outside the boundaries of the grid. We assume that $x_{i,j,k}$ equals zero if $i \notin \{1,\ldots,I\}$ or $j \notin \{1,\ldots,J\}$.

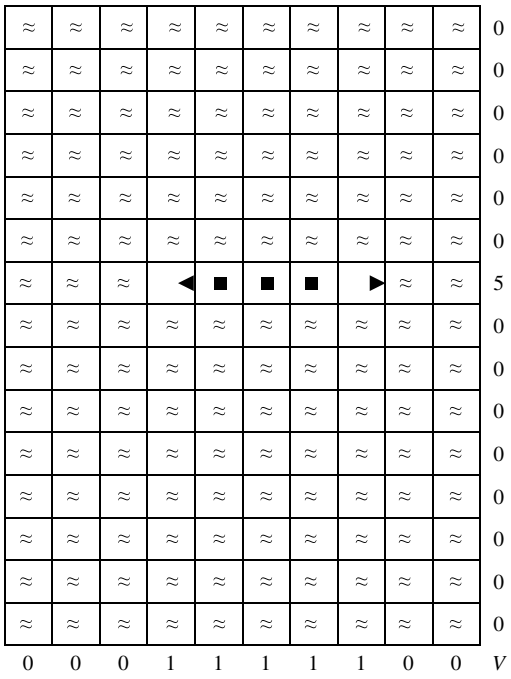
From constraint (1), it follows that each cell contains one out of seven symbols; that is, no cell can be empty. Subsequently, if a cell cannot contain a piece of a ship, it must contain water. Constraint (2) requires that if a symbol in a cell is given a priori, this symbol cannot be changed or accompanied by another symbol. Constraint (3) enforces that the sum of the cells that contain a piece of a ship in row i of the final grid is equal to the required number of cells that must contain a piece of a ship in row i. The same restriction holds for the columns and is given by constraint (4). From constraint (5), it follows that if cell (i,j) contains a ship of length 1, $s_{i,j,1}$ must be equal to one and zero otherwise. Constraint (6) requires that if cell (i,j) contains the first piece of a ship (i.e., symbol $\blacktriangleleft$ or $\blacktriangle$) of length $l \neq 1$, the decision variable $s_{i,j,l}$ is equal to one. Constraint (7) enforces that the sum of the ships of length l in the final grid is equal to the required number of ships of length l that has to be placed. The ships can be placed either horizontally or vertically. Combining this with the restriction that every ship has to be surrounded by water, it follows that it is not possible that two ships touch each other crosswise, constraints (8)–(9). Furthermore, each symbol that indicates a piece of a ship restricts the preceding and following cells. Those restrictions are given in constraints (10)–(21). Constraint (22) requires that submarines cannot touch each other horizontally or vertically. The other ships will not touch each other horizontally or vertically by definition of the crosswise restrictions (8)–(9). Finally, the variables $x_{i,j,k}$ and $s_{i,j,l}$ are restricted to be decision variables by constraints (24)–(25), respectively.

The code of the implementation of the cell-based model in AIMMS 3.6 can be found in the file "AIMMS code of cell-based IP model.aim."

3. Ship-Based Model

In this section, the *Battleship* puzzle is modelled by combining grids in which one ship is located. An

Figure 3 One of the Grids for the Ship-Based Model of Example 1



example of such a grid, used in the solution of Example 1, can be found in Figure 3.

There are five sorts of ships. Each of these ships can be placed in a grid. Combining these grids gives a solution to the *Battleship* puzzle. Which combinations should be used and can be obtained by using the clues and restrictions?

In this section, some additional sets, parameters, and variables are defined. Afterwards, these are used to formulate the cell-based model.

Sets

Note the following data:

- $Grids_l$ consists of the grids in which one ship of length l is located and is a subset of the overall set *gridsall*.
- *Gridsall* consists of all grids in which one ship is located where each grid is numbered. It is the union of the grids in $grids_l$ (i.e., $\bigcup_l grids_l$). A particular grid is denoted by g, where $g \in \{1, \ldots, N\}$; N denotes the last grid.

Parameters

Note the following data:

- $M_{i,j} = 1$, if cell (i, j) contains a piece of a ship in the initial grid;
  $= 0$, if cell (i, j) contains water in the initial grid;
  $= 100$, otherwise.
- $G_{i,j,g} = 1$, if cell (i, j) contains a piece of a ship in the grid numbered g;
  $= 0$, otherwise.
- $P_{i,j,g}$ is equal to one if cell (i, j), cell $(i, j+1)$, cell $(i+1, j)$, or cell $(i+1, j+1)$ of grid g contains a piece of a ship and is zero otherwise; that is,
- $P_{i,j,g} = 1$, if $G_{i,j,g} = 1$, $G_{i,j+1,g} = 1$, $G_{i+1,j,g} = 1$, or $G_{i+1,j+=1,g} = 1$;
  $= 0$, otherwise.

Variables

The decision variables are defined as follows:

- $y_g = 1$, if grid g is used in the final grid;
  0, otherwise.

Now a ship-based model can be formulated as a feasibility problem.

Constraints

$$\sum_g \sum_j y_{g_1} G_{i,j,g} = R_i \quad \forall i, \tag{26}$$

$$\sum_g \sum_i y_g G_{i,j,g} = C_i \quad \forall i, \tag{27}$$

$$\sum_{g \in grids_l} y_g = S_l \quad \forall l, \tag{28}$$

$$\sum_g P_{i,j,g} y_g \le 1 \quad \forall i, j, \tag{29}$$

$$\sum_g y_g G_{i,j,g} = M_{i,j} \quad \forall i, j \mid M_{i,j} = 0 \text{ or } M_{i,j} = 1, \tag{30}$$

$$y_g \in \{0, 1\} \quad \forall g. \tag{31}$$

Constraint (26) requires that the sum of the cells that contain a piece of a ship in row i is equal to the required number of cells that must contain a piece of a ship in row i. The same holds for the columns by constraint (27). Constraint (28) enforces that the sum of the ships of length l in the final grid is equal to the required number of ships of length l that has to be placed in the grid. Constraint (29) ensures that no grids are used in the final grid, which cannot act together. Two grids cannot act together if the ships touch each other, either horizontally, vertically, or crosswise. Moreover, two ships cannot overlap each other. Note that those grids that have ships that will overlap each other or touch each other in cell (i, j), or cell $(i, j+1)$, or cell $(i+1, j)$, or cell $(i+1, j+1)$ both have $P_{i,j,g} = 1$. From constraint (30), it follows that if the initial grid has a piece of a ship in a cell, the final grid also has a piece of a ship in that cell; if the initial grid has water in a cell, then the final grid also has water in that cell. Finally, the variable y_g is restricted to be a decision variable by constraint (31).

The code of the implementation of the ship-based model in AIMMS 3.6 can be found in the file "AIMMS code of ship-based IP model.aim."

4. Comparison of the Models

This section compares the complexity of the models presented in §§2 and 3.

The complexity of the cell-based model of §2 is, in part, determined by the number of variables and constraints. Recall that I is the number of rows and J is the number of columns. The total number of variables is at most equal to

$$12IJ + 1, \tag{32}$$

and the total number of constraints is at most equal to

$$29IJ - 2I - 2J + 5. \tag{33}$$

The total number of variables in the ship-based model is at most equal to

$$9IJ - 10I - 10J + 1 \tag{34}$$

for $I \geq 4$, $J \geq 4$ and at least one strict inequality. The total number of constraints is at most equal to

$$2IJ + I + J + 5. \tag{35}$$

Although the difference in the number of variables is relatively small, the number of constraints strongly differs. Because the ship-based model has fewer constraints than the cell-based model, it can be expected that puzzles will be solved faster by the ship-based model than by the cell-based model. However, it should be mentioned that the complexity of the models also depends on the structure of the constraint matrix. Some empirical results on this structure are discussed below when the models are used to solve Examples 1 and 2.

The models were implemented in the computer package *AIMMS* 3.6, in which the solver *CPLEX* 10.0 is used. The models are used to solve Example 1 of the introduction and Example 2 of the appendix. The solution times of Table 2 are obtained by a *Dell Inspiron Intel*(*R*) *Core*(*TM*)*2 Duo CPU* (*RAM 4 GB*). The table shows that the ship-based model has fewer variables, fewer constraints, smaller number of nonzeros, and solves faster than the cell-based model. Note that the cell-based model is not able to solve the puzzle of Example 2 within 10 hours of computation time.

Remark 2. For this paper two slightly different models with the technique of combining grids have been considered. The first model replaces the parameter $P_{i,j,g}$ by a parameter, which is equal to 1 if two grids cannot act together and is zero otherwise. Constraint (29) is replaced by a constraint, which excludes the use of two grids when this parameter is nonzero. The second model uses a slightly different formulation of $P_{i,j,g}$; that is, $P_{i,j,g}$ is equal to four if cell (i, j) contains a piece of a ship, is equal to one if one of the surrounding cells of (i, j) contains a piece of a ship, and is zero otherwise. By this definition, the summation in constraint (29) should be at most 4 instead of 1. Comparison of the models based on number of variables, number of nonzeros, and number of constraints showed that both models are less efficient than the ship-based model presented in this paper. The same conclusion can be drawn by comparison of the solution times found for the examples by each model.

5. Create a Puzzle with a Unique Solution

This section describes how a puzzle can be created. One of the main ingredients of a puzzle is the existence of a unique solution. By creating the puzzle, we therefore start with the solution of a puzzle that has been created manually; afterwards, we remove redundant information such that the solution of the puzzle remains unique. In this section, a method is presented to find unique solutions of a puzzle. This is done for the ship-based model. However, the same method can also be applied to the cell-based version of the model. At the end of this section, we use this method to create a large-size puzzle.

To determine whether a puzzle is unique, the model has to be run twice. Define the parameter $\bar{y}_g$ to be the solution to the puzzle, after the first run.

Table 2 Solution Times of Examples 1 and 2

Variables	Number of variables	Number of nonzeros	Number of constraints	Solution time	Number of iterations
Cell-based model					
Example 1	1,806	18,736	3,274	0.26 sec.	975
Example 2	7,206	110,049	11,539	>10 hours	>36,822,030
Ship-based model					
Example 1	1,101	15,856	220	0.00 sec.	0
Example 2	4,901	75,351	884	665 sec.	2,979,082

Now the uniqueness of the puzzle can be determined by introducing the following objective function:

$$\max -\sum_{g\,|\,\bar{y}_g=1} y_g + \sum_{g\,|\,\bar{y}_g=0} y_g.$$

Note that because the objective function is maximized, the term

$$-\sum_{g\,|\,\bar{y}_g=1} y_g$$

makes it optimal to set the value of y_g to zero when $\bar{y}_g = 1$, whenever that is feasible. On the other hand, the term

$$+\sum_{g\,|\,\bar{y}_g=0} y_g$$

makes it optimal to set the value of y_g to one when $\bar{y}_g = 0$, whenever that is feasible. Now if the puzzle is unique, the solution of the second run, y_g, will be equal to the solution of the first run, $\bar{y}_g$, so that the value of the objective function will become zero.

The puzzle of Example 1 has been tested for redundant information. Cells $(1, 5)$, $(1, 9)$, and $(7, 10)$ are found to be redundant. That means that the puzzle still has a unique solution even when these cells are removed from the initial grid. Finally, the technique described in this section has been used to create a puzzle: Example 2. The puzzle has a unique solution, but still contains redundant information.

Supplementary Material

Files that accompany this paper can be found and downloaded from http://ite.pubs.informs.org.

Appendix

Example 2. In most puzzle magazines, the size of the grids differ, where the smallest puzzle is of size 5×5 and

Table A.1 Specification of the Number of Ships of Each Sort That Has to Be Placed in the Grid of Example 2

Type of ship		Number to be placed
Submarine	●	23
Destroyer	◀ ▶	29
Cruiser	◀ ■ ▶	12
Battleship	◀ ■ ■ ▶	9
Carrier	◀ ■ ■ ■ ▶	7

Figure A.1 Initial Grid of Example 2

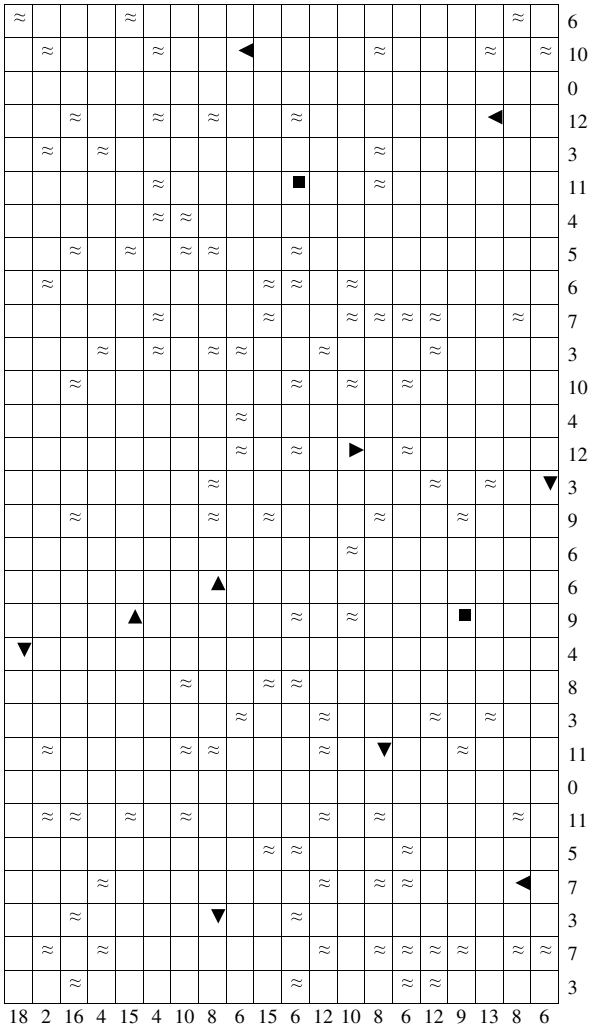
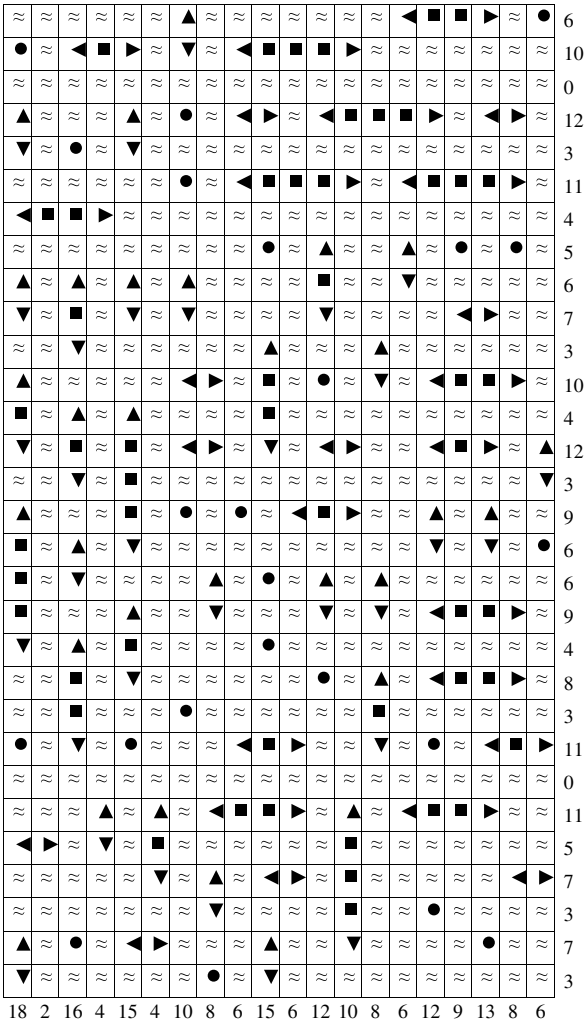


Figure A.2 Solution Example 2



the largest puzzle is of size 15×10. Figure A.1 and Table A.1 show a puzzle of size 30×20, which is created by using the technique described in §5. The solution of this puzzle is given in Figure A.2.

References

Battleship puzzle. 2009. Wikipedia (the free encyclopedia). Accessed July 31, 2007, http://en.wikipedia.org/wiki/Battleship_(puzzle).

Chen, S. T., S. S. Lin. 2004. Optimal algorithms for $2 \times n$ Mastermind games—Graph-partition approach. *Comput. J.* **47**(5) 602–611.

Chlond, M. J. 2005. Classroom exercises in IP modeling: Su Doku and the Log Pile. *INFORMS Trans. Ed.* **5**(2) 77–79.

Den Hertog, D., P. B. Hulshof. 2006. Solving Rummikub problems by integer linear programming. *Comput. J.* **49**(6) 665–669.

Gedeon, T. D. 1996. An iterative solution produced by transformation. *Comput. J.* **39**(4) 353–356.

Letavec, C., J. Ruggiero. 2002. The *n*-Queens problem. *INFORMS Trans. Ed.* **2**(3) 101–103.

Majumdar, A. A. K. 1994. A note on the iterative algorithm for the Reves puzzle. *Comput. J.* **37**(5) 463–464.

Sapir, A. 2004. The Tower of Hanoi with forbidden moves. *Comput. J.* **47**(1) 20–24.

Sniedovich, M. 2002. OR/MS games: 2. Tower of Hanoi. *INFORMS Trans. Ed.* **3**(1) 45–62.

Van de Liefvoort, A. 1992. An iterative algorithm for the Reves puzzle. *Comput. J.* **35**(1) 91–92.

Wilson, J. M. 1989. Crossword compilation using integer programming. *Comput. J.* **32**(3) 273–275.